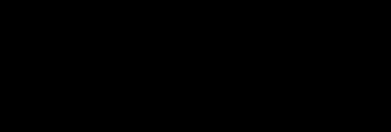


Building QuickTime-Savvy Applications

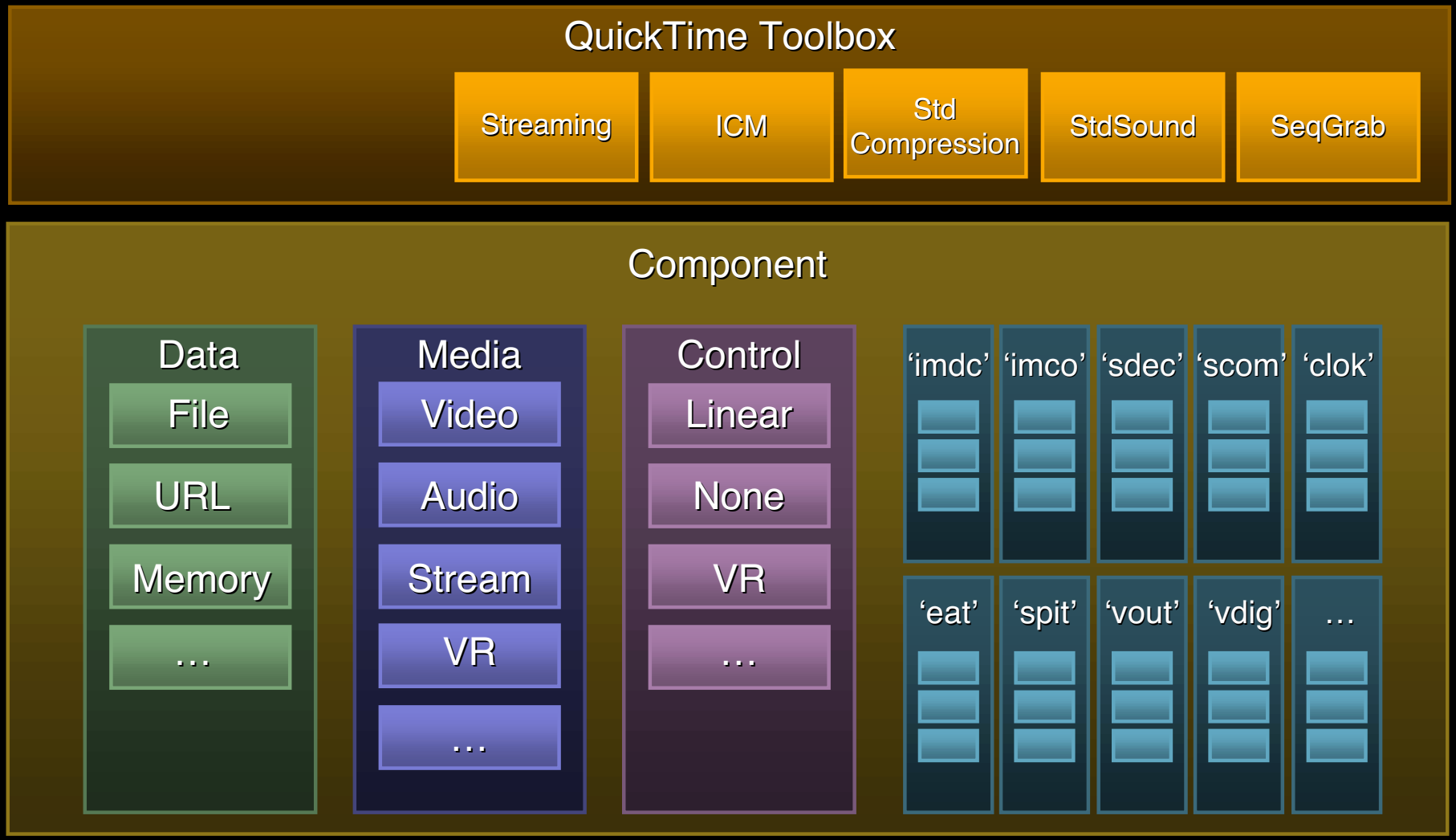
Session 601

Building QuickTime-Savvy Applications

Ian Ritchie
QuickTime Engineering

QuickTime?

QuickTime Overview



Topics Covered

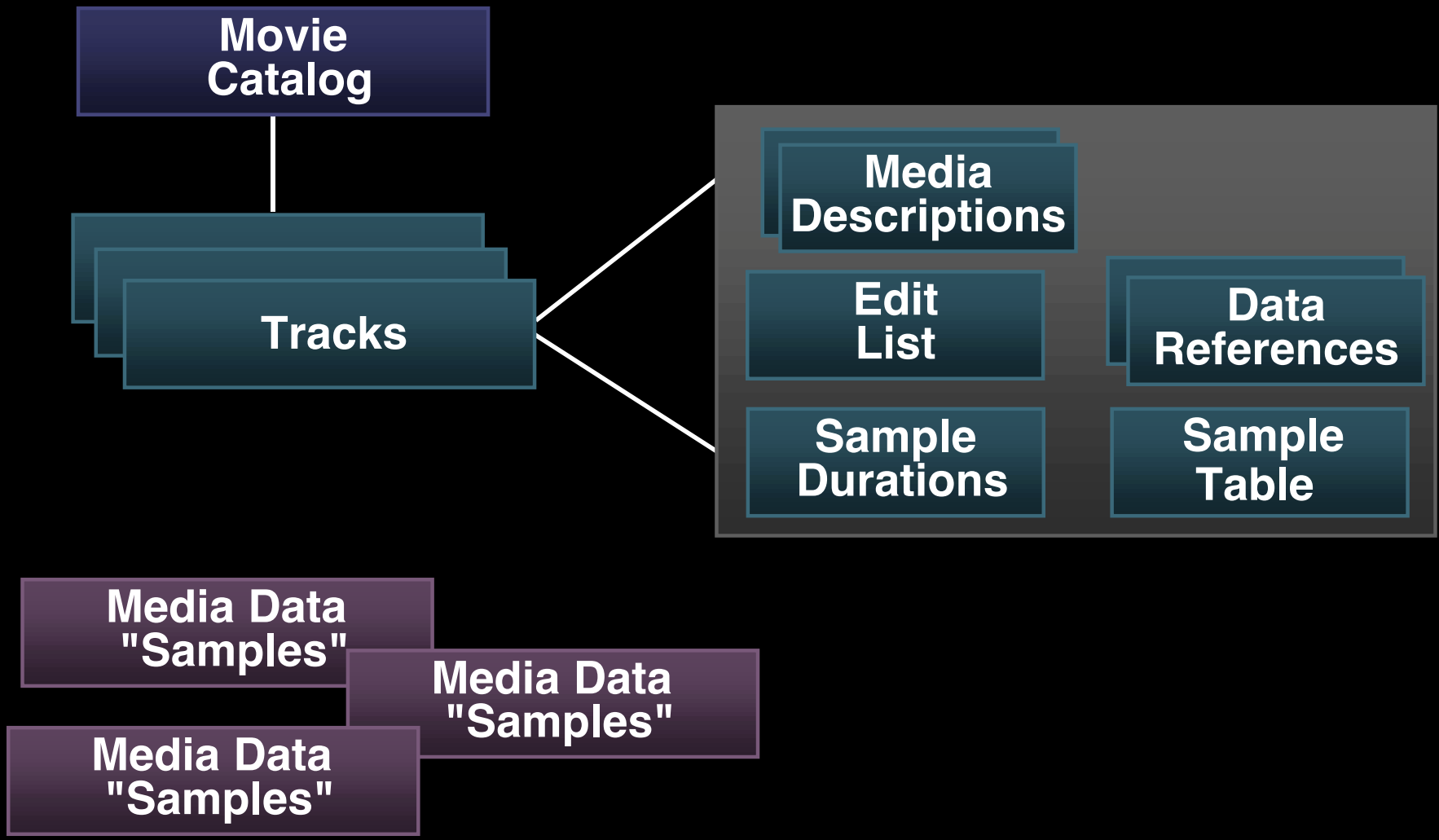
- Movies 101
- Images Import/Export
- Compression
- Data Handlers
- Video Processing
- QTVR
- Interactivity
- Mac OS X, Carbon
- Windows
- Carbon Movie Control
- Cocoa
- QuickTime Broadcaster

Movies 101

Brad Ford and Jon Summers
QuickTime Engineering

What Is a Movie?

Movie Is a Media Catalog

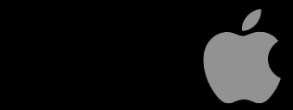


Demo

Movie Editing

Where Does Movie Data Come From?

- Local storage
  - Disk file (not necessarily a ".mov" file)
- Remote source
  - URL specified http:// rtsp://
- Memory
  - Handle, Pointer or Resource
- Different pieces can come from different places



Working With Movies

- **EnterMovies();**
  - Opening a movie
  - Tasking QuickTime
  - Movie playback
  - Editing
  - Saving
- Applications should *not* call **ExitMovies();**

DataReference

- Abstracted location specifier
- Handle to a data structure
- DataRefType specifies accessor component
- DataReference & DataRefType ~ DataHandler
- Local file DataReference is an AliasHandle

  QTNewAlias(&fsSpec, &dataRef, true);

  DataRefType ‘alis’ ~~ AliasDataHandler

Loading a Movie

NewMovieFromDataRef()

general form of **NewMovieFromFile()**

```
err = NewMovieFromDataRef (&movie, flags, 0,
                dataRef, dataRefType,);
```

Do not need to call **OpenMovieFile** or **CloseMovieFile**

Demo

FRONTBASE Data Handler

Movie Controllers

```
err = NewMovieFromDataRef (&movie, flags, 0,
      dataRef, dataRefType);
mc = NewMovieController (movie, &rect, mcFlags);

 // Run event loop. Task QuickTime with
MCIsPlayerEvent( )
// Your MovieController communicates with the Movie

// When you're finished with a movie...
DisposeMovieController (mc); // remove controller first!
DisposeMovie (movie);
```

MovieController ≠ MovieController Bar

- The Movie Controller bar is a convenience interface
- You can attach a Movie Controller to your movie and gain all of its functionality without showing the bar

Why Use Movie Controllers?

- They simplify control:

```
MCDoAction( mc, mcActionPrerollAndPlay,
                        (void *)myFixedRate );
MCDoAction( mc, mcActionGoToTime,
                        (void *)aTimeRecord );
MCDoAction( mc, mcSetKeysEnabled,
                        (void *)true );
MCDoAction( mc, mcActionLinkToURL,
                        (void *)myURLHandle );
```

Some Things Will Not Work Without Movie Controllers

- Interactivity
- QuickTime VR
- Streaming

They Let You Query Your Movie

- Get movie info

```
curTime = MCGetCurrentTime ( mc, &theTimeScale );

err = MCGetControllerInfo ( mc, &myFlags );

if (myFlags & mcInfoIsPlaying)

        // the movie is currently playing
```

Action Filters

```
MCSetActionFilter( mc, myActionFilterUPP);

Boolean myActionFilterCallback(
   MovieController mc, short action, void *params )
{
   if( action == mcActionMovieClick ) {
         (eventPtr)->what = nullEvent; // kill click
         return true; // I've handled this action
   }
   else
         return false; // handle this as usual
}
```

Tasking With Movie Controllers

- Give your movie processing time
- WaitNextEvent-based apps should call **MCIsPlayerEvent()**
  - New in QT6: **QTGetTimeUntilNextTask()**
  - More on this later

Support Movie Editing

```
MCEnableEditing( mc, true );

switch (editCmd) {
        case kCut:          MCCut (mc); break;
        case kCopy:         MCCopy (mc); break;
        case kPaste:        MCPaste (mc, srcMovie); break;
        case kClear:        MCClear (mc); break;
        case kUndo:         MCUndo (mc); break;
        case kTrim:         MCTrimMovieSegmentmc); break;
        case kAdd:          MCAddMovieSegment (mc, srcMovie,  scaled);
}
```

Adding Media Samples

- Adding compressed media data samples
  - **BeginMediaEdits()**
  - **AddMediaSample()**
  - **EndMediaEdits ()**
  - **InsertMediaIntoTrack ()**

New Movie From Scratch

```
CreateMovieStorage (dataRef, dataRefType,
    creator, 0, kDataHCanRead | kDataHCanWrite,
    &dataHandler, &movie);

// add tracks and media

AddMovieToStorage (movie, dataHandler);
CloseMovieStorage (dataHandler);
DisposeMovie (movie);
```

Support Long Unicode Names

- File Manager Based

FSSpec and file references

OpenMovieFile, CloseMovieFile
CreateMovieFile, DeleteMovieFie
AddMovieResource, UpdateMovieResource
PutMovieIntoDataFork64

- Movie Storage API

DataReference and DataHandler

OpenMovieStorage, CloseMovieStorage
CreateMovieStorage, DeleteMovieStorage
AddMovieToStorage, UpdateMovieInStorage
PutMovieIntoStorage

Saving the Movie

- Create or update catalog in storage

```
err = AddMovieToStorage ( movie, dataHandler );
err = UpdateMovieInStorage ( movie, dataHandler );
```

- Flattening

```
err = FlattenMovieDataToDataRef ( movie, flattenMovieFlags,
        dataRef, dataRefType, creator, 0, createMovieFlags );
```

- Exporting

```
err = ConvertMovieToFile ( movie, (Track)nil,
        (FSSpec*)nil, fileType, creator, 0, nil,
        exportMovieFlags,  (Component)nil );
```


Congratulations
Movies 101 Graduate
Now go write some
QuickTime 6 Savvy Applications
WWDC2002

Video Processing

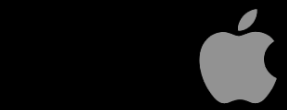
Tom Dowdy
QuickTime Engineering

More Video

- Capture
- Compression
- Import
- Export

Capture

- Sequence grabber
  - Audio
  - Video
- Source: HackTV



Compression

- Video processing
  - Other
  - Live video
  - Stored movie
  - Movie playback
  - Learn more: Session 602

Import/Export

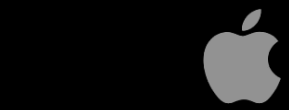
- Open file types
- Save as file types
- “Print to Video”
- UI or API to configure

Still Images

Sam Bushell
QuickTime Engineering

Still Images in QuickTime

- Graphics Importers
- Graphics Exporters
- Easy to use
- Opt-in for more powerful features



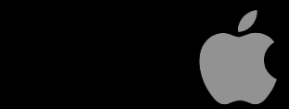
Drawing Still Image Files

```
GetGraphicsImporterForDataRef(
            dataRef, dataRefType, &gi );
GraphicsImportSetDestRect( gi, &rect );
GraphicsImportDraw( gi );
CloseComponent( gi );
```

- Draws BMP, FlashPix, GIF, JPEG, MacPaint, PDF (on Mac OS X), Photoshop, PICT, PNG, QTIF, TGA, TIFF, SGI, and others

Graphics Importers Also Support . . .

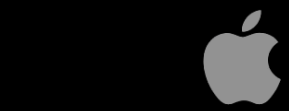
- Scale, rotate, perspective (improved in QT6)
- Transfer modes (blend, alpha composition)
- Clipping
- Reading metadata (e.g., Exif)
- Multiple images per file
  (Photoshop layers; TIFF pages; Exif thumbnails)
- Extensible by third parties (e.g., DICOM)



Writing Still Image Files

```
OpenADefaultComponent(
    GraphicsExportComponentType,
    kQTFileTypeJPEG, &ge );
GraphicsExportSetInputGWorld( ge, gw );
GraphicsExportSetOutputDataReference( ge,
    dataRef, dataRefType );
GraphicsExportDoExport( ge, nil );
CloseComponent( ge );
```

- Can write BMP, JPEG, MacPaint, Photoshop, PICT, PNG, QTIF, TGA, TIFF, SGI, and others



Graphics Exporters Also . . .

- Provide user dialogs for format-specific options
- Store metadata (Exif)
- Write image thumbnails (Exif)
- Extensible by third parties (e.g., PhotoJazz)

Demo

http://developer.apple.com/samplecode/Sample_Code/QuickTime/Basics/ImproveYourImage.htm

Working With Interactive Movies

Tim Monroe
QuickTime Engineering

Interactive Movies

- Respond to mouse, button, and keyboard events
- These events may trigger actions
- Two interactive track types
    - QuickTime VR tracks
    - Flash tracks

Interactive Movies

- Two “scriptable” track types
  - Text tracks
  - Sprite tracks
- You can add other interactive media types

Demo

Interactive Movies

Interactive Movies

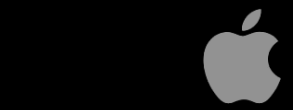
- Must be associated with a movie controller
- Wired objects can send actions to other objects in a movie or even in another movie

Demo

Intermovie Communication

Skinned Movies

- A skinned movie is a movie with a custom window shape
- No window frame is drawn and no controller bar is visible
- Allow the content creator to specify the entire look and feel of a movie, but . . .
    - ...must provide a means of controlling movie (e.g., a wired sprite controller)



Demo

Skinned Movies

Writing Cross-Platform QuickTime Applications

QuickTime and Windows

- QuickTime 3.0 provided identical APIs on Macintosh and Windows
- "Macintosh" data types are available on Windows, along with functions to create and dispose of instances of those types:

```
myBuffer = NewHandleClear(1024);
...
if (myBuffer != NULL)
        DisposeHandle(myBuffer);
```

QuickTime Media Layer

- Provides support for those portions of the Macintosh OS and UI Toolbox needed by QuickTime, including:
  - Memory Manager
  - File Manager
  - Gestalt Manager
  - Control Manager
  - Window Manager
  - Dialog Manager
  - Resource Manager

QuickTime Media Layer

- QTML is not a general-purpose porting layer
- QTML is a special-purpose porting layer;
  it is designed to allow developers to move
  QuickTime code to Windows quickly and easily

"Write once, deliver many . . ".

Porting Issues

- Namespace conflicts
- Endianness of multibyte data
- Resource data conversion
- Modeless dialog boxes

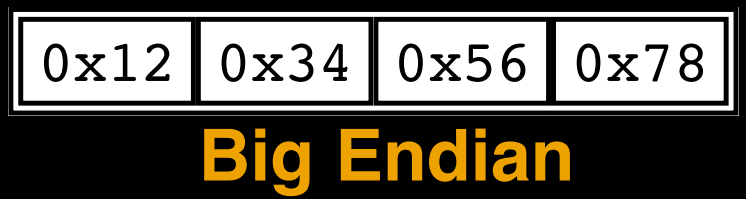
Namespace Conflicts

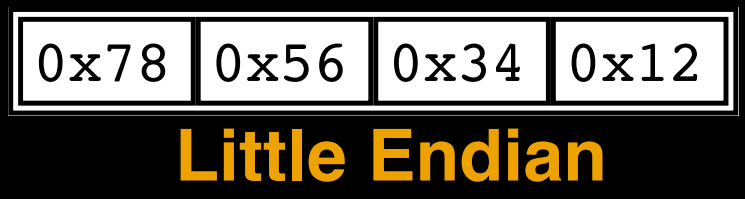
- Windows APIs overlap with some existing Mac APIs
- Mac APIs renamed to avoid this conflict:

  MacShowWindow
  MacSetPort
  MacInsertMenu
  MacOffsetRect

Endian Issues

- Data in QuickTime movie files is big-endian; Windows is little-endian





- In general, you work with native-endian data:
  - Parameters to QuickTime APIs
  - Data returned by QuickTime APIs

Endian Issues

- Sometimes data is returned in big-endian format

```
myUserData = GetMovieUserData(myMovie);
if (myUserData != NULL) {
    GetUserDataItem(myUserData, &myPoint, sizeof(Point),
        FOUR_CHAR_CODE('WLOC'), 0);
    myPoint.v = EndianS16_BtoN(myPoint.v);
    myPoint.h = EndianS16_BtoN(myPoint.h);
}
```

Endian Issues

- Byte swapping necessary only for multi-byte data
    - *But*, not required for C or Pascal strings
- Byte swapping is necessary only when data is transferred between RAM and some external container (e.g., a file)
- Use the macros!

Resource Data

- QTML supports Resource Manager
- On Windows, you need to explicitly open your application's resource fork:

```
myLength = GetModuleFileName(NULL, myName, MAX_PATH);
NativePathNameToFSSpec(myName, &mySpec,
        kFullNativePath);
gAppResFile = FSpOpenResFile(& mySpec, fsRdWrPerm)
UseResFile(gAppResFile );
```

Resource Data

- Resource Manager returns native-endian data for known resource types
- For custom resources, you can write a *resource flipper* and install it with the Resource Manager:

  RegisterResourceEndianFilter

- . . Or you can just flip the data yourself in your code

Modeless Dialogs

- On Windows, we usually do not call IsDialogEvent or DialogSelect
- Instead, install a callback function to handle Windows messages for a modeless dialog box

  SetModelessDialogCallbackProc

- . . Or, consider using native Windows APIs for complex dialog boxes

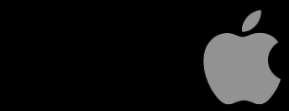
QuickTime and Carbon

- Carbon *is* a porting layer
- Some Carbon changes will affect our Windows code:

```
myID = (**myMenuHdl).menuID;          // pre-Carbon

myID = GetMenuID(myMenuHdl); // Carbon

// Windows or pre-Carbon Mac code
#if !ACCESSOR_CALLS_ARE_FUNCTIONS
#define GetMenuID(mHdl)                (** mHdl).menuID
#endif
```



Tasking QuickTime

Greg Chapman
QuickTime Engineering

Tasking QuickTime

- "Inside Macintosh: QuickTime" says you should task QuickTime
  - regularly . . .
  - as often as possible . . .
- There is a lot of folklore about this

There Has Got to Be a Better Way!

New in QuickTime 6

- An API you can use to find out when QuickTime needs to be tasked next
  - **WaitNextEvent**-based apps can call it to get the number of Ticks to pass to WNE
  - **CarbonEventLoopTimer**-based apps can call it to schedule a timer's next fire time
  - Working sample code available

But You Should
Not Have
To Do This

Higher Level Abstractions on Mac OS X

- For Cocoa:
  - **NSMovieView** (smarter with QT6)
- For Carbon:
  - **MovieControl** (new in QT6)

Carbon MovieControl

- Manages the playback and manipulation of QuickTime Movies
- Always creates a movie controller (can be hidden)
- Handles events
  - Mouse and keyboard events
  - Can participate in Edit menu and edit command handling
- The app can install event handlers, too

Carbon MovieControl

```
CreateMovieControl(theWindow, nil, theMovie, 0, &theControl);
RunApplicationEventLoop();
```

- That's all you have to do . . .
- Working sample code available

Demo

Carbon MovieControl

QuickTime Broadcaster

Adrian Baerlocher
QuickTime Engineering

QuickTime Broadcaster

- Cocoa Application using QuickTime
- Classes encapsulate QuickTime functional groups
    - Sequence Grabber
    - Standard Compression
    - Broadcast APIs

Demo

QuickTime Broadcaster

Roadmap

602 QuickTime for Video-Intensive Apps	Room A2 **Wed., 2:00pm**
603 Media Integration With QuickTime	Room A2 **Wed., 3:30pm**
604 Delivering Content via Interactive QuickTime	Room A2 **Wed., 5:00pm**
605 Developing QuickTime Components	Room A2 **Fri., 9:00am**

Roadmap

606 QuickTime for the Web	Room A2 **Fri., 2:00pm**
607 QuickTime and MPEG-4: A Technical Overview	Room A2 **Fri., 3:30pm**
812 QuickTime Streaming Server 4	Civic **Thurs., 2:00pm**
FF010 QuickTime	Room J1 **Fri., 10:30am**

Who to Contact

Developer Technical Support
dts@apple.com

Jeff Lowe
QuickTime Technology Evangelist
jefflowe@apple.com

http://developer.apple.com/wwdc2002/urls.html

For More Information

- QuickTime online documentation
  developer.apple.com/quicktime
- QuickTime sample code
  developer.apple.com/samplecode/Sample_Code/QuickTime

Reminder

The QuickTime Engineering Team
Is Holding a “Hands-On Lab” Everyday
From 1:00–4:00pm in Room G . . Stop By!


WWDC2002


WWDC2002


WWDC2002